Ateneo de Manila University

IPC: Signals

Department of Information Systems and Computer Science
S.Y. 2001-2002
http://sysads.ateneo.net/wyu/
wyu@admu.edu.ph
Interprocess Communications

★ mechanism in which processes communicate with one another

★ in order to facilitate communication among different processes

★ special mechanisms for process communications:
  – signals
  – semaphores
  – shared memory
  – pipes
  – message passing
Signals

★ are software generated interrupts

★ can also be generated when an error occurs such as SIGFPE (floating point exception) and SIGSEGV (segfault)

★ can also be generated when a hardware event such as a bus error or an illegal instruction is encountered

★ are pre-specified by the system and defined in signal.h
Classes of Signals

- Hardware
- Software
- Input/output
- Process control
- Resource control
Signals

Behavior when a signal is received by a process:

★ signal is discarded after being received
★ process is terminated after the signal is received
★ a core file is written, then the process is terminated
★ process is suspended after the signal is received
Common Signals

⋆ are number from 1 to 31

⋆ common signals defined in signal.h
  - SIGHUP 1 hangup
  - SIGINT 2 interrupt
  - SIGQUIT 3 quit
  - SIGILL 4 illegal instruction
  - SIGABRT 6 used by abort
  - SIGKILL 9 hard kill
  - SIGALRM 14 alarm clock
  - SIGTERM 15 soft kill
  - SIGCONT 19 continue a stopped process
  - SIGCHLD 20 to parent on child stop or exit
Sending Signals

* int kill(pid_t pid, int sig);
* int raise (int sig);

  - sends sig signal to the processes pid
  - if pid is zero then it refers to all processes in the process group
  - if pid equals -1, then sig is sent to every process except for the first one
  - if pid is less than -1, then sig is sent to every process in the process group
  - if sig is zero no signal is sent
  - raise () is equivalent to kill(getpid(), sig)
  - returns -1 when an error occurs
Signal Handling

★ signals can be caught and its behavior can be changed

★ what can happen after catching a signal?
   – process can let the default action happen
   – process can block the signal (some signals cannot be ignored)
   – process can catch the signal with a handler

★ Signal handlers
   – usually execute on the current stack of the process
   – can be changed on a per-signal basis
   – a process must resume in a different context than the interrupted one, it must restore the previous context itself
Signal Handling

⋆ void (*signal(int signum, void (*sighandler)(int)))(int);

– installs a new signal handler sighandler for the signal with number signum

– sighandler can access either of the following:
  ⋆ SIG_IGN causes the signal to be ignored
  ⋆ SIG_DFL causes the signal to execute its default action
  ⋆ a pointer to a function

– signals SIGKILL and SIGSTOP cannot be caught or ignored
## Default Signal Actions

<table>
<thead>
<tr>
<th>Signal</th>
<th>Number</th>
<th>Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIGHUP</td>
<td>1</td>
<td>A</td>
<td>death of controlling process/terminal</td>
</tr>
<tr>
<td>SIGINT</td>
<td>2</td>
<td>A</td>
<td>interrupt from keyboard (ctrl-c)</td>
</tr>
<tr>
<td>SIGQUIT</td>
<td>3</td>
<td>C</td>
<td>quit from keyboard</td>
</tr>
<tr>
<td>SIGILL</td>
<td>4</td>
<td>C</td>
<td>illegal instruction</td>
</tr>
<tr>
<td>SIGABRT</td>
<td>6</td>
<td>C</td>
<td>abort signal from abort(3)</td>
</tr>
<tr>
<td>SIGFPE</td>
<td>8</td>
<td>C</td>
<td>floating point exception</td>
</tr>
<tr>
<td>SIGKILL</td>
<td>9</td>
<td>AEF</td>
<td>kill signal</td>
</tr>
<tr>
<td>SIGSEGV</td>
<td>11</td>
<td>C</td>
<td>invalid memory reference</td>
</tr>
<tr>
<td>SIGPIPE</td>
<td>13</td>
<td>A</td>
<td>broken pipe</td>
</tr>
</tbody>
</table>

Figure 1: Default Signal Actions
<table>
<thead>
<tr>
<th>Signal</th>
<th>Number</th>
<th>Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIGALRM</td>
<td>14</td>
<td>A</td>
<td>timer signal from alarm(2)</td>
</tr>
<tr>
<td>SIGTERM</td>
<td>15</td>
<td>A</td>
<td>termination signal</td>
</tr>
<tr>
<td>SIGUSR1</td>
<td>30,10,16</td>
<td>A</td>
<td>User-defined signal 1</td>
</tr>
<tr>
<td>SIGUSR2</td>
<td>31,12,17</td>
<td>A</td>
<td>User-defined signal 2</td>
</tr>
<tr>
<td>SIGCHLD</td>
<td>20,17,18</td>
<td>B</td>
<td>child stopped or terminated</td>
</tr>
<tr>
<td>SIGCONT</td>
<td>19,18,25</td>
<td>DEF</td>
<td>continue if stopped</td>
</tr>
<tr>
<td>SIGSTOP</td>
<td>17,19,23</td>
<td>DEF</td>
<td>stop process</td>
</tr>
<tr>
<td>SIGTSTP</td>
<td>18,20,24</td>
<td>D</td>
<td>stop typed at tty</td>
</tr>
<tr>
<td>SIGTTIN</td>
<td>21,21,26</td>
<td>D</td>
<td>tty input for background process</td>
</tr>
<tr>
<td>SIGTTOU</td>
<td>22,22,27</td>
<td>D</td>
<td>tty output for background process</td>
</tr>
</tbody>
</table>

Figure 2: Default Signal Actions
Signal Actions:

A  terminate process
B  ignore signal
C  terminate process and dump core
D  stop/suspend the process
E  signal cannot be caught
F  signal cannot be ignored